

VirHKey: a VIRtual Hyperbolic KEYboard with gesture interaction and visual feedback for mobile devices

Benoît Martin

LITA, University of Metz

Île du Saulcy

57045 Metz, Cedex 01, France

33 3 87 54 73 17

benoit.martin@univ-metz.fr

ABSTRACT

The *VirHKey* is a new text entry method for mobile devices using a simple *Unistroke*-like alphabet. Gesture recognition is accomplished not through pattern recognition but through sequences of flicks with low required angular accuracy: $2\pi/5$ in radians. This means that the full stroke path is unimportant and recognition is highly deterministic, enabling good accuracy. All during the interaction, an optional visual feedback is provided at least for the learning stage. This is done with a focus and context display based on the hyperbolic geometry. To illustrate the possibilities of the *VirHKey*, a prototype application was developed and usability tests demonstrated the overall good performances of the proposition with a text entry rate from 18 to 25 words per minute and a high satisfaction of users.

Categories and Subject Descriptors

H.5.2 [User Interfaces]: Evaluation/methodology, Input devices and strategies, Interaction styles.

General Terms

Performance, Design, Experimentation, Human Factors.

Keywords

Focus and context, hyperbolic geometry, pen-based interaction, gesture keyboard.

1. INTRODUCTION

With small mobile devices, classical keyboards are difficult or even impossible to use. In this case, software keyboards give many alternative solutions [10]. We can distinguish several kinds of software keyboards from the interaction technique point of view. In this paper, we focus on gesture keyboards: the interaction is done by gestures with a pen and each gesture is interpreted as a character or a command. Usually, the recognition is based on the Latin alphabet or a dedicated one. In the first case, the recognition

is difficult and many errors can occur. Thus, the shapes of characters are often simplified but gestures remain complicated and user must use a gesture sheet to memorize them at least at the learning stage. In the second case, a new alphabet is proposed. Gestures can be simpler as in *Unistroke* proposition to improve speed and reduce errors during recognition but users must invest the required effort to learn this new alphabet. A visual feedback helps sometimes user to do gestures as in *QuikWriting* proposition but the gestures become more complicated and it is difficult not to use the visual feedback. Thus, even if these keyboards have proved to be natural and efficient in some classes of tasks [10], there are many situations where the users have difficulties to learn it or to use it.

We propose a new concept of gesture keyboard as simple as the *Unistroke* one while providing a visual feedback. This truly optional visual feedback helps user to learn progressively and swiftly the alphabet at the learning stage. In next sections, we present the existing propositions in order to show their possibilities and their limits. Among the many encountered difficulties, it is necessary to take into account the space available to visualise the keyboard, the learning time and the accuracy which is needed in gesture interactions. Then we present the *VirHKey* project from the hyperbolic tool to the gesture interaction techniques passing by the characters layout. Next, the usability tests are described followed by the results. Finally, we present our conclusion.

2. GESTURE KEYBOARDS

In gesture keyboards, the interaction is mainly done with a pen on a tactile surface. We can distinguish two categories: the symbolic keyboards and the target keyboards.

2.1 Symbolic keyboards

A symbolic keyboard uses gesture recognition to associate a gesture to one and only one character. Unfortunately, the Latin alphabet remains complicated: it needs always high computation capacities and good accuracy in the gesture. Each symbolic keyboard introduces a simplified alphabet. With *Unistroke*, a character is obtained with only one gesture [7]. The alphabet was designed for the simplicity to ease the learning, to minimize the ambiguity between gestures and to improve speed. The most frequent characters are associated to the simplest gestures (figure 1). No comparative study has been undertaken but an idealistic text entry rate of 34 wpm (words per minute) was reported.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobileHCI'05, September 19–22, 2005, Salzburg, Austria.

Copyright 2005 ACM 1-59593-089-2/05/0009...\$5.00.



Figure 1. Samples of *Unistroke* and *Graffiti 2* alphabets.

Graffiti was introduced by Palm Computing for its PDAs [1]. The majority of the characters are obtained with only one gesture but not all as the 'X' (figure 1). It was clearly designed to reduce the learning time: the gestures are close to the Latin alphabet but they are much more complicated than the *Unistroke* ones. [6] reported a text entry rate from 7 wpm (novice) to 21 wpm (expert) and reported a 9% errors rate for both novice and expert users. At this time *Graffiti* is a success and users appreciate the ease of learning even to the prejudice of performance. Thus the new gesture keyboards have now to show their ease to learning in comparison to *Graffiti*.

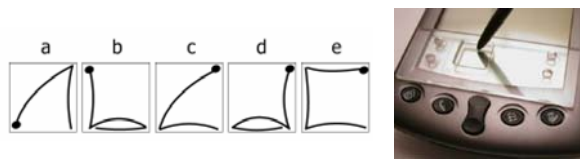


Figure 2. Samples of *EdgeWrite* alphabet.

The last keyboard described here is *EdgeWrite*. It was conceived to simplify the recognition and to need lower accuracy from the user [16]. Its alphabet is *Unistroke* and it was designed from the order by which the user passes in the four corners of a square. The gesture looks like the corresponding Latin character (figure 2). The errors are reduced because the interface is tangible: gestures are done in a seizure guide which constraints the pen. [16] reported a 18% better accuracy than *Graffiti* for novice users with no significant difference in speed.

2.2 Target keyboards

A target keyboard uses gestures to reach a target or a zone assigned to a character. All the characters are presented to the user but if the display is incomplete, the characters are distributed in different pages. A visual feedback is provided but after few hours of practice, the visual feedback can become unnecessary. From the interaction point of view, we distinguish the static selection and the dynamic selection.

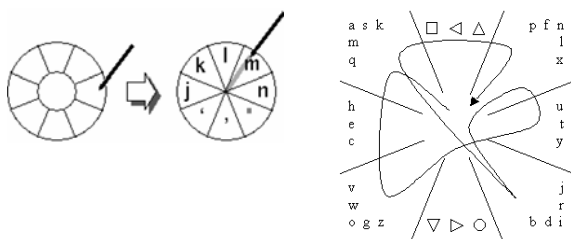


Figure 3. The layouts of *T-Cube* and *Quikwriting*.

With the static selection, the gesture associated to a character is always the same. For example, *T-Cube* [15] uses hierarchic pie-menus (figure 3). The gesture, a flick, uses a departure point and a direction. The departure point is done in one of the first pie-menu to indicate which sub pie-menu is concerned. The direction indicates which character to select in the sub pie-menu. The gestures are simple but an accuracy of $2\pi/8$ in radians is required. [15] indicated that reasonably fast text entry can be achieved. One

subject achieved 21 wpm but it appears difficult to learn. *Quikwriting* works with continuous gestures on a 3×3 grid [13]. Characters are entered with gestures that begin in the centre cell (figure 3). User moves to the cell which contains the desired character. If the desired character is at the centre of the cell, it is sufficient to go back directly to the centre cell (for example 'i'). On the contrary, if the user goes to one of the two adjacent cells before to go back to centre cell, he obtains the next character in the direction of the chosen adjacent cell (for example 'u'). Similarly, in the corners which contain 5 characters, user can go in the second adjacent cells before to go back in the centre cell to obtain the second next character in the direction of the chosen adjacent cell (for example 'k'). Letters that occur more frequently have the shortest gestures which can be done with various devices. With a pen, [8] reported a text entry rate always improving of 17 wpm (expert) with errors correction.

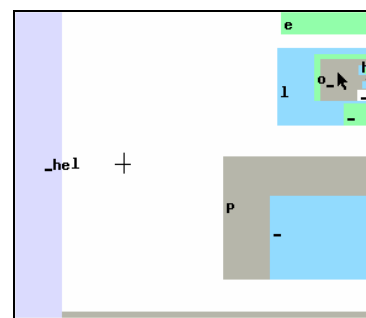


Figure 4. The *Dasher* layout.

With the dynamic selection, the gesture which is associated to a character varies according to the state of the keyboard. A visual feedback is necessary. For example, in *Dasher* [15], the user points where he wants to go and the display zooms where he points. The world is painted with characters, so that any point he zooms in corresponds to a piece of text. The user chooses what to write by choosing where to zoom. Figure 4 shows the writing of "hello how are you". Of course, it requires sustained visual attention from the user. For efficiency, *Dasher* uses prediction rules and [9] reported text entry rate from 25 wpm (novice) to 35 wpm (expert) with a mouse.

3. THE VirHKey PROJECT

The *VirHKey* project for VIRTUAL Hyperbolic KEYboard uses the pentagonal tiling of the hyperbolic space. Its aim is to provide an alphabet as simple and efficient as *Unistroke* while helping user with a visual feedback at the learning stage. To be easily adapted to mobile devices, the visual feedback is based on the focus and context technique to use little space on screen. In next sections, we present the hyperbolic tool of the *TYPHOON* project which is used as the basis of the *VirHKey*. Then, we propose a characters layout before to describe the gesture technique.

3.1 Hyperbolic tool of the TYPHOON project

In the *TYPHOON* project (Tiling of the hYperbolic Plane for Human-cOmputer inter-actiON) we developed a generic focus and context tool by using the hyperbolic geometry. Even if this technique has been already often used, it was never by using the tiling of the hyperbolic plane.

3.1.1 Hyperbolic geometry and the tiling

Hyperbolic geometry is the result of replacing the parallel axiom of Euclidean geometry with the alternative of axiom that through a given point there are at least two lines parallel to a given line. We shall use Poincaré's disc model: the open unit disc U of the Euclidean plane constitutes the points of the hyperbolic plane. Lines are diameters or arcs of circles which are orthogonal to the border of U . The distortion of the mapping between the infinite hyperbolic plane and the open disc gives a good way to visualise data. A regular tiling is a splitting of the space with a unique generator polygon by translation or by reflection in sides to cover the entire space. Figure 5 shows the pentagrid which is generated from a central regular pentagon with an interior angle of $\pi/2$ in radians.

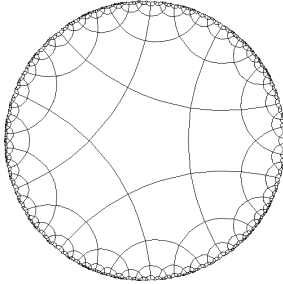


Figure 5. The pentagrid.

The hyperbolic plane is split into basic regions R_i generated by the reflection of the central polygon into the common sides. This splitting generates a tree: nodes are pentagons and arcs are the reflections. This tree is a spanning tree of the neighbourhood graph: additional connections are represented in bold and dotted lines (figure 6).

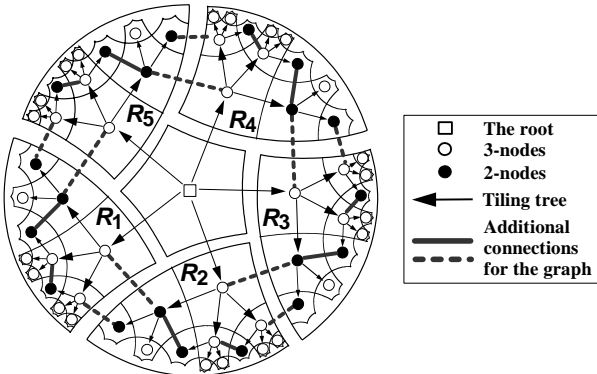


Figure 6. The tiling tree and the neighbourhood graph.

As it was first shown [11], each pentagon of a region can be identified by a unique number from the standard Fibonacci numbering. By adding the number of the region, we number all pentagons with a unique couple $w : r$ with r the region and w the Fibonacci number in this region. By extension, the central polygon is $0 : 0$. From this numbering, we can compute the neighbourhood of all pentagons and explore the tiling without additional data structures. Details can be found in [3] [4] [5] [12].

3.1.2 Hyperbolic tool for visualisation

The tiling can be used to visualize and to explore various types of data. The main difficulty is to find how to distribute data in the

tiling. We note T the mapping function as $T(w, r)$ gives the data associated to pentagon $w : r$. The first technique to visualise the data is the absolute visualisation: the data associated to pentagon $w : r$ is displayed in pentagon $w : r$. Figure 7 shows the result: the data is just the number of the polygon and it is displayed by " $w:r$ " with w in decimal. In specific applications, data will be displayed differently: text, colour, etc.

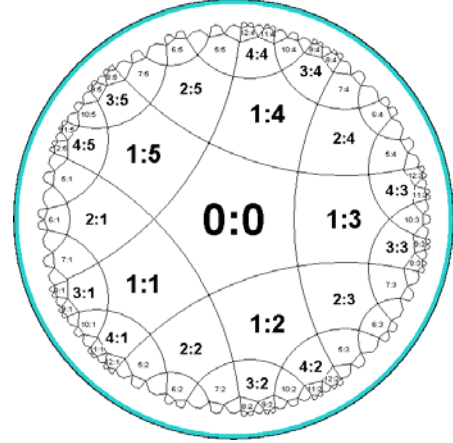


Figure 7. Absolute visualisation (depth = 4).

On one hand, the hyperbolic tool gives a focus to the data displayed in the central pentagon $0 : 0$. On the other hand, the tool gives the context with all data surrounding the central one. Then, the second technique we propose is the relative visualisation: we dissociated the pentagons of the display (the receiver) from the data which is displayed in. A data associated to a pentagon $w' : r'$ can be displayed in a pentagon $w : r$. Of course the neighbourhood is kept. By bringing a data to the centre of the tiling, the user focuses on it. The distortion of the mapping gives the feeling of a zoom because the central pentagon is larger. In figure 8, user focuses on the data of pentagon $2 : 1$. Some data not yet visible in figure 7 appear and some visible data disappear.

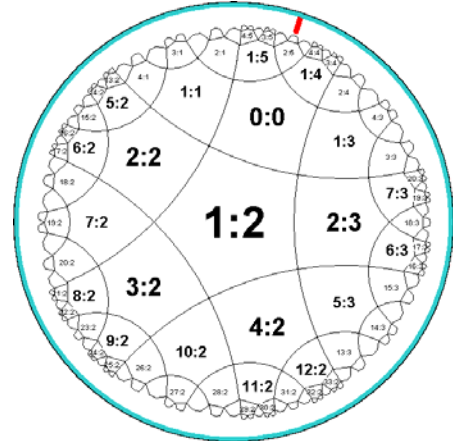


Figure 8. relative visualisation (depth = 4).

Various algorithms were developed to manage data [3] [4]. In particular, we provided the zooming algorithm to bring a data on top of region R_{src} in the central pentagon and to bring the central data in the pentagon on top of region R_{dest} . We denote this algorithm by $\{src::dest\}$. For example, figure 8 is obtained by applying zooming $\{2::4\}$ to figure 7.

3.2 The characters layout

The idea is to consider a pentagon as a key and to consider a character as the data of a pentagon. We call layout, the assignment of characters to the keys. A layout is desirable if it is both predictable and fast. With a predictable layout, a moderately experienced user can guess at the interactions to make a desired character, and be right some of the time. In a fast layout, frequent characters correspond to fast access. A predictable layout helps the novice user and a fast layout helps the expert user.

The clustering of the characters is crucial for a predictable layout. One of the most frequently used characters, the space, is associated to the central pentagon (□ in figure 9). The predictable clustering of the characters was designed according to the vowels. This reveals a convenient and natural clustering of the characters:

Let $L = \{L_k\}$ with $k \in \{1..26\}$
 $= \{ \text{ABCDEFGHIJKLMN} \text{OPQRSTUVWXYZ} \}$

and $V = \{V_k\}$ with $k \in \{1..5\}$
 $= \{ \text{AEIOU} \}$

The clusters C_i containing characters are defined by:

$C_i = \{L_k\}$ as $L_k \in [V_i, V_{i+1}[$ with $i \in \{1..4\}, k \in \{1..26\}$
and $C_5 = \{L_k\}$ as $L_k \geq V_4$ with $k \in \{1..26\}$

Thus, the whole set C of clusters is defined by:

$C = \{C_k\}$ with $k \in \{1..5\}$
 $= \{ \text{ABCD, EFGH, IJKLMN, OPQRST, UVWXYZ} \}$

Cluster C_i with $i \in \{1..5\}$ defines the characters of region R_i of our hyperbolic tool. The first character of C_i , the vowel, is grounded in the first pentagon of region R_i . The three next characters are grounded counter clockwise in the three pentagons of the next level. The process goes on from the fourth character if there are yet some characters in cluster C_i .

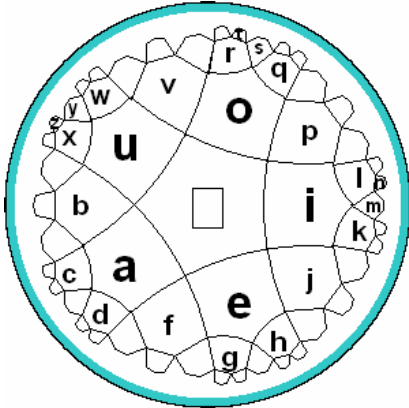


Figure 9. Characters layout in the pentagrid.

Figure 9 shows the pentagrid with this layout. Several possible layouts were considered for this design but they are a myriad of possible layouts. Future work may discover a better layout.

3.3 The gesture interaction

The focus and context aspect of the *VirHKey* involves a hierarchical exploration: the user interacts only with the five first pentagons of the regions. In this section we propose a fast, self disclosing *Unistroke* alphabet. Since a “flick” gesture is fast with

a pen and easy to recognize, the alphabet of *VirHKey* consists of flicks as in *T-Cube* [15]. A flick has two important aspects, the starting point and the direction. The direction can be horizontal or diagonal, specifying the character of one of the five first pentagons (figure 10). The required accuracy, $2\pi/5$ in radians, is lower than the $2\pi/8$ of *T-Cube*. This is easier for users.

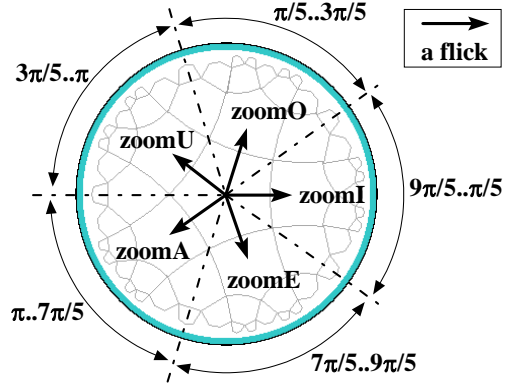


Figure 10. The five flicks in the *VirHKey*.

A flick is complete when a target typically 0.4 to 0.7 inches in diameter (depending to the setup) is reached. The corresponding zoom is triggered in the keyboard. We note Z the set of these possible zooms:

$Z = \{Z_i\}$ with $i \in \{1..5\}$
 $= \{ \text{zoomA, zoomE, zoomI, zoomO, zoomU} \}$

zoomA is the zoom to ground the character which is on top of region R_1 in the central pentagon. And so on for the other zooms. For example, a flick between $\pi/5$ and $3\pi/5$ will trigger a zoomO. Any interactive system needs to give feedback to help the user understand, learn and use it. *VirHKey* enlists both optional visual and audio feedbacks. It gives feedback of the running flick which is displayed with a red edge directly on the keyboard (figure 11).

When a flick is complete, the corresponding zoom is triggered and the *VirHKey* is modified thanks to *zooming* algorithm according to table 1. A sound can be played for a brief time too.

Table 1. Mapping flick and zooming in the pentagrid.

zoomA	zoomE	zoomI	zoomO	zoomU
{1::3}	{2::4}	{3::5}	{4::1}	{5::2}

From the initial state of the *VirHKey*, the first flick gives only access to the five vowels. More flicks are required for the other characters. A path P is a sequence of zooms to reach a character:

Let $P = (z_1, z_2, \dots, z_n)_{n \in \mathbb{N}}$ with $z_i \in Z$.

Figure 11 shows the two flicks to reach character ‘q’ and the visual cue provided to user. Thus, a gesture G_q attached to character ‘q’ is $f \rightarrow$ which corresponds to the path $P_q = (\text{zoomO, zoomI})$. The gestures can be done on a separate touch sensitive surface (for example a tablet) or directly on the screen if it is touch sensitive (for example the screen of a TabletPC or a PDA). In this last case, the gesture has not to start in the disc of the *VirHKey*. The flicks are displayed in the centre of the *VirHKey* (figure 11) but user can do the gesture $f \rightarrow$ where he wants on the screen: the spatial position of the pen compared to the position of

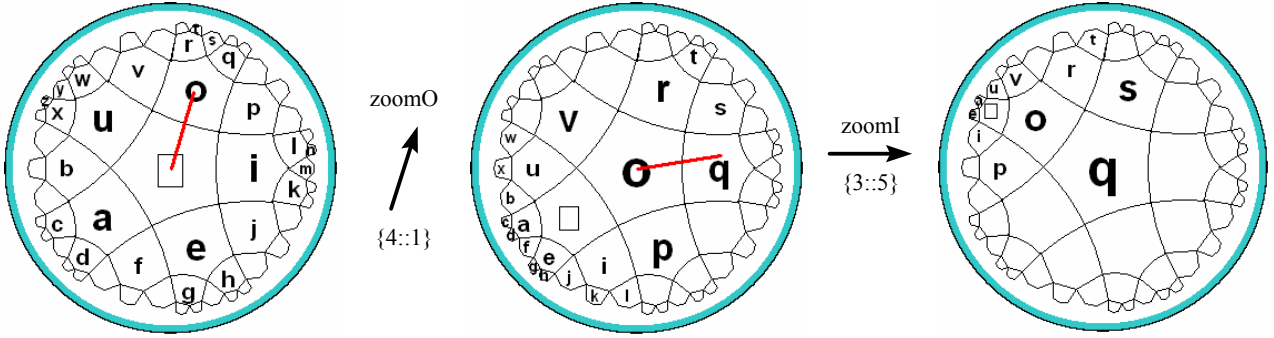


Figure 11. Visual feedback for the 'q' character.

the *VirHKey* has no importance for the gesture computation. The direction is the only important parameters. When the pen is lifted, the validation is automatically triggered. Then, the current character in the central pentagon is "typed" to the system and the *VirHKey* returns to the initial state for the next gesture.

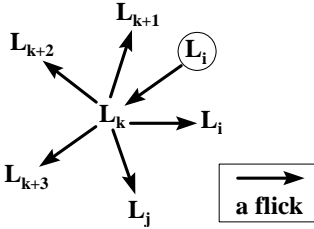


Fig. 12. The gesture properties of *VirHKey*.

User can predict the gestures thanks to the characters layout. Figure 12 shows a typical situation: user comes from character L_i and applied a flick to focus on character L_k . Then, if this character is not the right one, he can use the following properties to reach the next character:

- The first flick in the counter clockwise direction with respect to the last one grounds the next character, L_{k+1} in the central pentagon. The second and the third flick in the counter clockwise direction ground L_{k+2} and L_{k+3} respectively.
- The first flick in the clockwise direction grounds the previous character, L_i in the central pentagon. It is the cancellation flick.
- The second flick in the clockwise direction has no simple property. It is the alternative flick and the visual cue is needed in this case.

Unfortunately, the cancellation flick is not so simple to implement. For example, zoomA is cancelled by zoomI but, according to table 1, zoomA is $\{1::3\}$ and zoomI is $\{3::5\}$. The character in the central pentagon will be right but not the other ones modulo a rotation of $2\pi/5$. For example, the character which was on top of region R_1 should be on top of region R_5 . We must apply the zooms by taking into account the last one. In the previous example, the zoom is a zoomI but as the previous one was zoomA, it will correspond to a $\{3::1\}$ denoted by zoomA^{-1} instead of $\{3::5\}$. More generally, if $z \in Z$ and $z = \{src::dest\}$ then $z^{-1} = \{dest::src\}$. Table 2 shows the rules to interpret a zoom according to the previous one.

Table 2. Contextual interpretation of the zooms.

new previous	zoomA	zoomE	zoomI	zoomO	zoomU
\emptyset	zoomA	zoomE	zoomI	zoomO	zoomU
zoomA	zoomA	zoomE	zoomA^{-1}	zoomO	zoomU
zoomE	zoomA	zoomE	zoomI	zoomE^{-1}	zoomU
zoomI	zoomA	zoomE	zoomI	zoomO	zoomI^{-1}
zoomO	zoomO^{-1}	zoomE	zoomI	zoomO	zoomU
zoomU	zoomA	zoomU^{-1}	zoomI	zoomO	zoomU

To limit errors during gestures and to increase speed, the *VirHKey* takes only into account flicks which lead to a character. Thus, we avoid empty pentagons. For example, in the gesture \nearrow , the last flick will not be triggered because the pentagon on the right of character 'q' is empty (figure 11). So, this gesture remains to gesture \nearrow of the character 'q'. The user is able to do ampler gestures to reach quickly characters even with suppose complicated gestures.

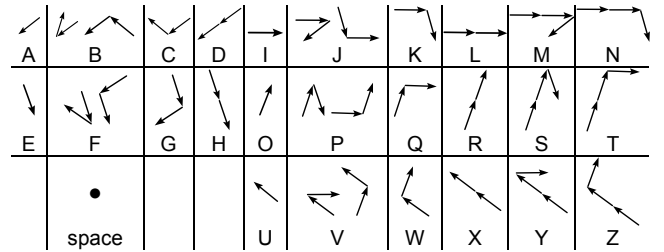


Figure 13. The standard alphabet of *VirHKey*.

For one character, the path is not unique but we have uniqueness if we do not consider path with alternative or cancellation flicks. Figure 13 shows this standard alphabet. At the opposite of *Graffiti* and co. there is no similarity between a gesture and the normal hand-printed character. This alphabet is *Unistroke*: one character is done with one gesture and each gesture has, at maximum, one shift of direction.

The feedbacks are flick dependant: the gestures are analyzed flick by flick and not just at the end of the gesture. The user can refer to the visual feedbacks but if he remembers the gesture, he may make it directly without any helps. We expect that the feedbacks will help the user to learn swiftly the alphabet before to become unnecessary.

4. USABILITY TESTS

To evaluate the *VirHKey*, we are doing a usability study using the protocol described in [9]. The first goal is to show its immediate

usability: many consumers, discouraged by their initial experience and frustration, may never invest the required effort to become experts. Next, we try to exhibit the users' improvements in entry speed and error rates from novice to expert level with a longitudinal study.

4.1 Participants

In longitudinal studies, fewer participants are usually engaged but over a long period of time. In this study, 5 participants from 22 to 47 years old were involved, all being native French speakers. They all use desktop computers on a regular basis and they were well informed on the time commitment required for the experiment.

4.2 Task and procedure

The task was done during 20 sessions and each session lasted about 20 minutes. Each session contained several blocks of trials and each block contained 10 text phrases of about 25 characters each. These 10 phrases were randomly selected from a source file of 70 French sentences. They were not repeated within a block but repeats were allowed between blocks. Each participant completed 20 sessions. Sessions were separated by at least two hours but no more than two days. This was to simulate a regular use of the system while trying to avoid fatigue. This was a longitudinal study attempting to practice participants toward expert performance. Data collection included numerous measurements on user input. For each key tapped, the following data were collected: given Character and position in the phrase, user entered character, elapsed time (ms) from previous tap and error (1 if the user character was wrong; 0 if the user character was correct)

The participants were asked to copy each phrase. The keyboard was in a 300x300 dots window and the audio and visual feedbacks were allowed. When an error occurred a prominent "click" was heard. They were asked to ignore errors and to carry on with the next correct character pointed at by the cursor. Typical experiment display is shown in figure 14 (left).

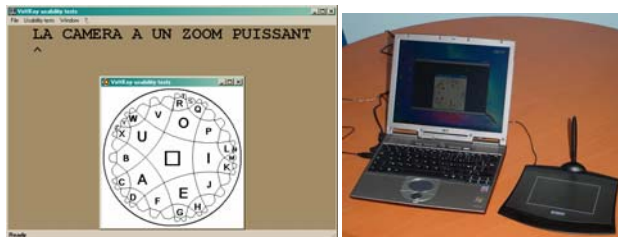


Figure 14. Experiment screen and apparatus.

All participants were first given instructions explaining the goal of the experiment and the task. They were asked specifically to aim for both entry speed and accuracy. Participants were constantly reminded to do their best.

4.3 Apparatus

The host system was an ACER TravelMate 370 with an Intel Centrino 1,5 Ghz processor running Microsoft Windows XP. The software was written in C++ under Micro-soft Visual C++. The display was a 12" CRT with 1024x768 pixels. The gestures were

done on a separate tactile surface, a Wacom Volito (figure 14, right).

4.4 Results and Discussion

Although the task happens continuously, there is a small break in text entry when a new phrase is presented. The participants tend to read the phrase for a while and begin writing only after comprehending the phrase or at least the first few characters. We excluded the data on the first characters.

The analysis of variance of text entry speed showed a significant effect of session ($F_{19,40} = 15.15, p < .001$). A Student test showed a significant speed difference between Nicolas and Kamel ($t = -4.12, p\text{-value} < .0003$). For novice, the average entry text speed is 6.60 wpm. This appears 5.8% slower than Graffiti for example [6]. This is not so bad because the alphabet is far from Latin one and must be learned. So, this result appears not as bad as it could have been. The average text entry rate for the *VirHKey* keyboard rises to 22.89 wpm by the 20th session. In average, the *VirHKey* is for example 9% faster than *Graffiti* [6]. Unfortunately, it is far from *Unistroke* idealistic speed. But after about 7 hours of practice, all participants had not become "experts" and the speed is always increasing. However, we expect that it will increase only few.

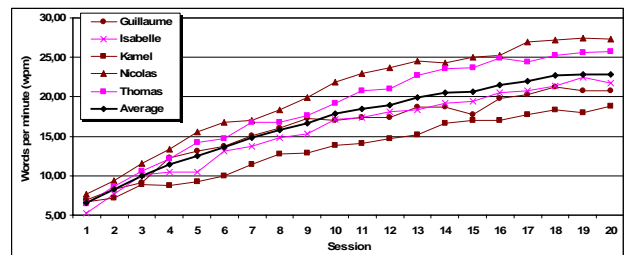


Figure 15. Text entry speed by session.

An error was recorded when the user-entered character differed from the given character. An analysis of variance revealed no significant effect of session. This may have occurred because participants controlled their speed to limit their errors. The average error rate ranged from 2.80% for the 1st session to 5.87% for the 20th session. Thus, at the beginning, the average error rate is 69% better than *Graffiti* [6] and even better than *EdgeWrite* in spite of his seizure guide. In 16th session, the *VirHKey* presents only an error rate of 5.81% at 21.52 wpm the maximum speed of *Graffiti*. This is always 35% lower than the *Graffiti* error rate at this speed. Moreover, this error rate stays constant until the maximum speed in the 20th session. As for the text entry speed, a Student test showed a significant difference between Nicolas and Kamel ($t = -3.59, p\text{-value} < .002$). The faster participant, Nicolas, did significantly more errors than the slower one, Kamel.

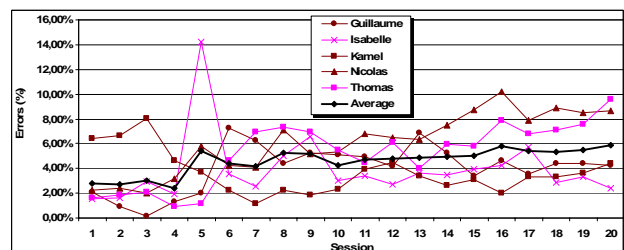


Figure 16. Error rate by session.

These results show very good performances both for speed and error rate. A more detailed analysis is actually done to measure user performances for each character. Figure 17 shows the speed entry rate for each character.

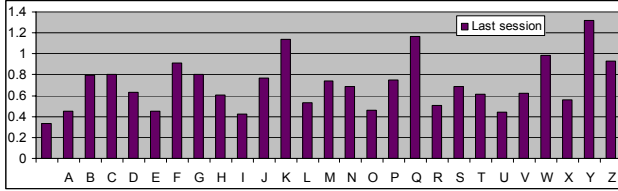


Figure 17. Time in seconds for each character.

We can notice that the space gesture (the first in the X-axis) is quicker than any other gestures. Moreover, we can note that there are no significant differences for A, E, I, O and U gestures. The gestures with one flick are quicker than the gesture with two or three flicks. This is not very surprising. But, surprisingly, gestures with three flicks are similar in time to the gestures with two flicks. This is an important point: the position of a character in the tiling is not so important to do quick gestures except for the gestures with one flick. The bad performances for the K, Q, Y and Z characters can be explained by the low frequency of these characters in French language.

Each participant filled a SUS-like questionnaire at the end of the 20 sessions [2]. The agreement with each question is expressed from 1 to 5 (1 = strongly disagree, 5 = totally agree). Then we can compute a SU value in the range 0 and 100. Greater the value is, more the participant is satisfied. We obtained 71.50 for the average SU value which indicates a good satisfaction of the participants (see table 3).

Table 3. SU-like questionnaire for the satisfaction of user.

	Guillaume	Isabelle	Kamel	Nicolas	Thomas
I think I would like to use the <i>VirHKey</i> frequently	3	3	3	2	4
I found the <i>VirHKey</i> unnecessarily complex	2	2	2	2	3
I think the <i>VirHKey</i> was easy to use	3	3	4	4	4
I think that I would need the support of a technical person to use the <i>VirHKey</i>	1	1	2	1	1
I would imagine that most people would learn to use the <i>VirHKey</i> quickly	5	4	3	3	5
I found the gestures of the <i>VirHKey</i> too hard to make	2	1	3	2	2
I felt very confident using the <i>VirHKey</i>	4	5	3	4	4
I needed to practice a lot before to could get going with the <i>VirHKey</i>	2	2	3	3	2
I am satisfied with the ease of entering characters with the <i>VirHKey</i>	2	5	2	4	4
I am satisfied by the speed to enter characters with the <i>VirHKey</i>	3	5	4	5	5
SU value	67.5	82.5	57.5	70	80

First analyze shows that the participants were surprised by their performance in speed and by the ease of using. They were surprised by their learning curve too because they were sceptical at the beginning of the experiment: they felt no frustration. This point is very important for a symbolic keyboard with gestures far from the Latin alphabet on the contrary of Graffiti for example. They both inquired a better characters layout and improvements of the visual feedback. In particular, to avoid bounce, they all asked to remove the character from the central pentagon. This should reduce a lot the error rate because space character is the main source of errors. This is due to shaking when the pen touches the tactile surface. This improvement will be tested soon.

5. CONCLUSION

We proposed a new virtual keyboard using a gesture text entry method. The gestures are based on a static selection with flicks similarly to *T-Cube* but with a required angular accuracy of $2\pi/5$ in radians instead of $2\pi/8$. The user interacts with an alphabet as simple as the *Unistroke*: each character is associated to one gesture which has, at maximum, one shift of direction. It guarantees swift interactions. The interaction can be done on a separate device or directly on the screen with a TabletPC or a PDA.

On one hand, the *VirHKey* guides the novice user to learn the gestures with a focus and context visual feedback using little space on screen. We expect that thanks to this help, users will get over the initial complexity. On the other hand, the *VirHKey* allows high speeds for expert users. First usability tests confirm the superiority of the *VirHKey* compared to most other gesture keyboards as *Graffiti*, *QuikWriting* or *T-Cube*. They show quick learning and a text entry rate from 6.5 wpm (novice) to 25.8 wpm (expert) with an average error rate from 3.21% (novice) to 5.48% (expert). The qualitative evaluation shows the satisfaction of the users and the lack of frustration.

6. PERSPECTIVES

We are implementing the *VirHKey* on a PDA in VGA mode (figure 18, left). Then, usability tests will be conducted in mobile situation and more precise data analysis will be done too to measure how the movement and interruption of mobile use affect the system's performance. Future work may measure usability without the visual feedback and propose some improvements. In particular, as we need no graphical relationship between a character and a gesture, we will try to find the best layout to match the most frequent characters with the easiest gestures to make. Unfortunately, these new layouts will be language dependant and no longer predictable. The new layouts will have to deal with other characters too: we only proposed to ground the letters but we have now to deal with numbers, punctuation, etc. We can for example add characters in the empty pentagons. Figure 18 (right) shows in light gray the available pentagons reachable with simple gestures. We can also use several discs as in the *QuikWriting* proposition, one for each purpose: accents, numerical characters, punctuation, uppercase and lowercase. The switch between discs can be associated to a pentagon ie. a gesture.

Moreover, the *VirHKey* is not limited to the gesture interaction technique. We will propose soon other interaction techniques like a reduced physical keyboard. More details will be given in a



	w	1	2	3	4	5	6	7	8	9	10	11
r												
l	a	b	c	d								
2	e	f	g	h								
3	i	j	k	l							m	n
4	o	p	q	r							s	t
5	u	v	w	x							y	z

Figure 18. The PDA implementation and the complete alphabet of the *VirHKey*.

forthcoming paper. Finally, more efficient way for training should be considered.

7. ACKNOWLEDGEMENTS

The author gratefully acknowledges M. Margenstern for helpfully hyperbolic discussions. Special thanks to X. Richez for his work on bibliography. This work was partly supported by the MICOLE European project (IST-2003-51152).

8. REFERENCES

- [1] Blickenstorfer, C. H. Graffiti: Wow!!!! Pen Computing Magazine. January, pp. 30-31, 1995.
- [2] Brooke, J. SUS - A quick and dirty usability scale. <http://www.cee.hw.ac.uk/~ph/sus.html>
- [3] Chelghoum K., Margenstern M., Martin B. and Pecci I. Cellular Automata in the pentagrid of the hyperbolic plane: tools for interactions. Publications of LITA, technical report, 2004-101, 63 pages, January 2004.
- [4] Chelghoum, K., Margenstern, M., Martin, B. and Pecci, I. Tools for implementing cellular automata in grid $\{7, 3\}$ of the hyperbolic plane. DMCS, Turku, Finland, July, 2004.
- [5] Chelghoum, K., Margenstern, M., Martin, B. and Pecci, I. Cellular automata in the hyper-bolic plane: proposal for a new environment. ACRI, Amsterdam, The Netherlands, October 25-27, 2004.
- [6] Fleetwood, M. D., Byrne, M. D., Centgraf, P., Dudziak, K. Q., Lin, B. and Mogilev, D. An Evaluation of Text-Entry in Palm OF Graffiti and the Virtual Keyboard. Proceedings of the HFES 46th Annual Meeting, pp. 617-621, <http://chil.rice.edu/fleet/documents/HFES02palm.pdf>.
- [7] Goldberg, D. and Richardson, C. Touch-typing with a stylus. Proceedings of INTERCHI'93, New York, pp. 80-87, 1993.
- [8] Isokoski, P. and Raisamo, R. Quikwriting as a Multi-device Text Entry Method. Pro-ceedings of NordiCHI 2004, ACM Press, Tampere, pp. 105-108, 2004.
- [9] MacKenzie, I. S., and Zhang, S. X. The Design and Evaluation of a High-Performance Soft Keyboard. Proceedings of CHI'99, New York, pp. 25-31, 1999.
- [10] MacKenzie, I. S., and Soukoreff, R. W. Text Entry for Mobile Computing: Models and Methods, theory and Practice. Human Computer Interaction, 17, pp. 147-198, 2002.
- [11] Margenstern M. New tools for Cellular Automata in the Hyperbolic Plane. Journal of Universal Computer Science, Vol. 6, issue 12 (2000), pp.1226-1252.
- [12] Margenstern M. Cellular automata in the hyperbolic plane. Publications of GIFM, technical report, 99-103, ISBN 2-9511539-6-1, 34 pages, December 1999.
- [13] Perlin, K. Quikwriting: Continuous stylus-based text entry. Proceedings of UIST'98. New York: ACM, pp. 215-216, 1998.
- [14] Venolia, D. and Neiberg, F. T-Cube: A fast, self disclosing pen-based alphabet. Proceedings of CHI'94, New York, pp. 265-270, 1994.
- [15] Ward, D. J., Blackwell, A. F. and MacKay, D. J. C. Dasher - A data entry interface using continuous gestures and language models. Proceedings of UIST 2000. New York: ACM, pp. 129-137, 2000.
- [16] Wobbrock, J.O., Myers, B.A., and Kembel, J.A. EdgeWrite: A Stylus-Based Text Entry Method Designed for High Accuracy and Stability of Motion. Proceedings of UIST'03, Vancouver, B.C., November, pp. 61-70, 2003.